Parallelization of Concurrent Processes in Higher Dimensional Automata(Theory of Rewriting Systems and Its Applications)

AUTHOR(S):

Takayama, Yukihide

数理解析研究所講究録
918巻 1995年 238-252

Parallelization of Concurrent Processes in Higher Dimensional Automata

Yukihide Takayama
Department of Computer Science, Ritsumeikan University
1916 Noji-cho, Kusatsu-shi, Shiga 525, Japan
email: takayama@theory.cs.ritsumei.ac.jp

July 24, 1995

Abstract

In many process algebras, a nondeterministic choice in particular form can be translated to a parallel composition with the expansion law. This translation, which we will call *parallelization*, is not trivial because we must find subprocesses in rather complex pattern in a given process. This paper gives a parallelization algorithm. The algorithm has a clear geometric meaning, but the precise definition needs Bar construction in cohomology theory of groups.

1 Introduction

Consider the simple process algebra with parallel composition $(P \mid Q)$, nondeterministic choice $P + Q$ (or $\Sigma_i P_i$), sequential composition $P\ ;\ Q$ and label restriction $P\backslash L$ over the set $\mathcal{A}$ of atomic actions. We can give the operational semantics as a labeled transition system and we will describe the transition as $P \xrightarrow{\alpha} Q$ $(\alpha \in \mathcal{A})$. Also, we can define the (strong) bisimulation relation $\sim$ with the transition. This kind of process algebra has the expansion law:

$$(P_1 \mid \cdots \mid P_n) \sim \Sigma\{\alpha\ ;\ (P_1 \mid \cdots \mid Q_i \mid \cdots \mid P_n) \mid P_i \xrightarrow{\alpha} Q_i, \alpha \in \mathcal{A}\}$$

The law indicates that the nondeterministic processes in some particular pattern can be translated to the more efficient processes with true concurrency. Then, we can consider the rewrite rule: $\Sigma\{\alpha\ ;\ (P_1 \mid \cdots \mid Q_i \mid \cdots \mid P_n) \mid P_i \xrightarrow{\alpha} Q_i\} \longrightarrow (P_1 \mid \cdots \mid P_n)$. Applying the rewrite rule to a given process seems to be rather complicated because we must find the complex pattern of nondeterminacy in the given process. We will refer to this job *parallelization* in this paper. The parallelization algorithm in our sense does not seem to be studied extensively so far.

The aim of this paper is to give a parallelization algorithm that has clear geometric meaning. The basic idea is to view parallelization as cycle filling procedure in the HDA (higher dimensional automata) model [3, 1]. We interpret concurrent processes in the domain of CW complexes [4]. Topological properties of the CW complex represent the nature of the concurrent processes. In this model, the parallelization algorithm is described as follows. First, we find suitable branches in the CW complex. Link the end points of the branches to obtain cycles. As branches may be higher dimensional, the cycles may also be higher dimensional. Next we fill all the cycles with suitable cells. Then the obtained CW complex represents the parallelized process. Finally, we carry out the *reverse interpretation* to extract the desired parallelized process expression from the CW complex.

For the precise description of this simple idea, we need the language in algebraic topology. In particular, the cycle filling procedure can be clearly described with Bar construction procedure in cohomology theory of groups [2]. Bar construction is applicable to simplical complexes, while the HDA model needs cubical complexes. To fill the gap between the simplical and the cubical

complexes, we must define a unit square as a collection of simplexes. That makes our HDA model a little more complicated than the HDA models such as those given by E. Goubault and T. Jensen [1]. On the other hand, the definition of the reverse interpretation has a feature of combinatorial geometry rather than algebraic topology. The procedure analyzes the geometric configuration of the cells in the given HDA space, finds subprocesses and combine them to form a process expression.

The configuration of the paper is as follows. Section 2 defines our process algebra and overview the idea of the HDA model. The formalization of the HDA model in terms of bar expression is given in section 3. The bar expression is the language for Bar construction. Section 4 discusses how we can find the suitable branches whose end points will be linked. Section 5 presents the cycle filling procedure in terms of Bar construction. The reverse interpretation procedure will be given in section 6 and the concluding remark will be given in the final section.

2 Process Algebra and its HDA model

2.1 Process language and expansion law

We will consider a simple process algebra with parallel composition, sequential composition, nondeterministic choice and label restriction. Let $\mathcal{L}$ be an enumerable set of of labels. $\overline{\mathcal{L}} = \{\bar{a} \mid a \in \mathcal{L}\}$ is called the set of *co-labels*. We assume that $\bar{\bar{a}} = a$ for any $a \in \mathcal{L} \cup \bar{\mathcal{L}}$. We also define the set of atomic actions $\mathcal{A} \stackrel{\text{def}}{=} \mathcal{L} \cup \overline{\mathcal{L}} \cup \{\tau_a \mid a \in \mathcal{L}\}$. The syntax of the process expressions, or simply processes, is defined as follows.

$$
\begin{aligned}
P \;:=\; & \mathit{nil}\ [\textbf{nil action}] \mid \alpha(\in \mathcal{A})\ [\textbf{atomic action}] \mid \tau_a\ [\tau\textbf{-action}] \\
& \mid (P \mid P)\ [\textbf{parallel composition}] \mid P + P\ (\text{or } \Sigma_i Pi)\ [\textbf{nondeterministic choice}] \\
& \mid P\ ;\ P\ [\textbf{sequential composition}] \mid P\backslash L\ [\textbf{restriction}].
\end{aligned}
$$

where $L(\subset \mathcal{L})$ is a finite set. We can give the operational semantics of the process language as a labeled transition system as usual, and the strong bisimulation relation $\sim$ is also defined as expected. We also have the expansion law: Let P_i $(1 \leq i \leq n)$ be process expressions. Then,

$$
\begin{aligned}
& (P_1 \mid \cdots \mid P_n)\backslash L \\
& \sim \Sigma\{\alpha\ ;\ (P_1 \mid \cdots \mid Q_i \mid \cdots \mid P_n)\backslash L \mid P_i \stackrel{\alpha}{\to} Q_i, \alpha \notin L \cup \bar{L}\} \\
& \quad + \Sigma\left\{\tau_a\ ;\ (P_1 \mid \cdots \mid Q_i \mid \cdots \mid Q_j \mid \cdots \mid P_n)\backslash L \,\middle|\, i \neq j, P_i \stackrel{a}{\to} Q_i, P_j \stackrel{\bar{a}}{\to} Q_j, a \in L\right\}
\end{aligned}
$$

We view the law as the rewrite rule from the right hand side to the left hand side, and call the rewriting with the rule *parallelization.*

Now we will define a few more notions. If a process P has a sequence of transitions $P = P_0 \stackrel{\alpha_1}{\to} P_1 \stackrel{\alpha_2}{\to} \cdots \stackrel{\alpha_n}{\to} P_n = \mathit{nil}$, we say that the process has a *computation path* $p = (\alpha_1, \alpha_2, \cdots, \alpha_n)$. We denote the set of all the computation paths of P by $cp(P)$. For a process P, *alphabet of the process* P, $\chi[P](\subset \mathcal{A})$, is defined inductively as follows:

1. $\chi[\mathit{nil}] = \phi$ and $\chi[\alpha] = \{\alpha\}$ if $\alpha \in \mathcal{A}$;
2. If P and Q are processes, then $\chi[(P \mid Q)] = \chi[P + Q] = \chi[P\ ;\ Q] = \chi[P] \cup \chi[Q]$.
3. If P is a process and $L(\subset \mathcal{L})$ be a finite set, then $\chi[P\backslash L] = \chi[P] - L \cup \bar{L}$.

2.2 HDA model of the process language

The basic idea of the HDA model is to represent a concurrent computation in a CW complex with a suitable direction. In the CW complex, a connected path represents a computation path and branches from a single point represent a nondeterministic choice. This is compatible with the naive idea of representing concurrent computations in ordinary directed graphs. We can find the

sharp difference from the graph based semantics in the representation of parallel composition. A true concurrent computation of n-atomic actions is represented by an n-dimensional unit square (n-*cell*). For example, consider a process $P = (a \mid b)$ $(a, b \in \mathcal{A},\ a \neq \bar{b})$. The HDA interpretation of P is a 2-cell illustrated in Figure 1 (a). $cp(P) = \{(a,b),(b,a)\}$ and its elements are described

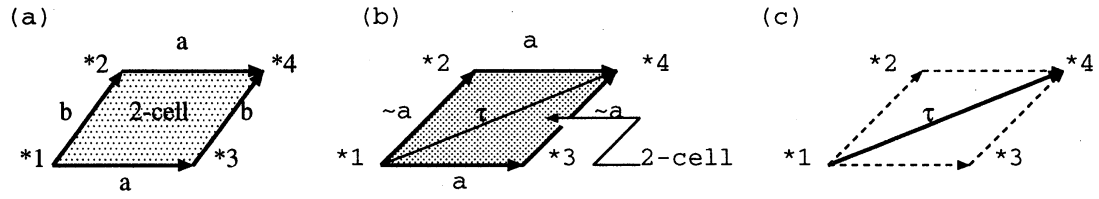


Figure 1: HDA spaces

by the paths $p_1 = *_1 \xrightarrow{a} *_3 \xrightarrow{b} *_4$ and $p_2 = *_1 \xrightarrow{b} *_2 \xrightarrow{a} *_4$ in the cell. The initial and the final states of P are represented by the points $*_1$ and $*_4$. Notice that the two paths are *homotopic*: they can be continuously transformed to each other through the 2-cell. Namely, the actions a and b are commutative. With this topological property, we regard that the 2-cell represents the situation in which the actions a and b runs truly concurrently.

This simple model can easily be refined to adopt the case of synchronous communication. Consider the process $Q = (a \mid \bar{a})$ $(a \in \mathcal{L} \cup \bar{\mathcal{L}})$. $cp(Q) = \{(a,\bar{a}),(\bar{a},a),(\tau_a)\}$. Hence, the HDA interpretation of Q is as in Figure 1 (b). The 1-cell τ_a represents the synchronous communication of the actions a and $\bar{a}$, and τ_a is contained in the 2-cell.

Next, we consider a process: $Q\backslash\{a\}$. $cp(Q\backslash L) = \{(\tau_a)\}$. The HDA interpretation is obtained by removing the 2-cell and the 1-cells with labels a and $\bar{a}$ from the HDA interpretation of Q. This is illustrated in Figure 1 (c).

Based on the idea given above, we define the HDA interpretation $[\![\cdot]\!]$ of our process algebra.

[HDA interpretation of the processes]

Empty action *nil*: $[\![nil]\!] = \phi$.

Atomic action: $[\![\alpha]\!]$ $(\alpha \in \mathcal{A})$ is the directed line segment $i_\alpha \xrightarrow{\alpha} f_\alpha$ where i_α and f_α are the initial and the final states of the action.

Parallel composition: $[\![(P \mid Q)]\!] = [\![P]\!] \times [\![Q]\!]$. Let i_P and i_Q be the initial states of P and Q, and let F_P and F_Q be the set of final states of P and Q. Then $i_{(P|Q)} =< i_P, i_Q > \in [\![(P \mid Q)]\!]$ and $F_{(P|Q)} = F_P \times F_Q \subset [\![(P \mid Q)]\!]$.

Nondeterministic choice: $[\![P + Q]\!] = [\![P]\!] \cup [\![Q]\!]$, and we assume that $[\![P]\!] \cap [\![Q]\!] = \{i_{P+Q}\}$. The set of final states is $F_{P+Q} = F_P \cup F_Q$.

Sequential composition: $[\![P\ ;\ Q]\!]$ is constructed as follows: Let $F_P = \{f_1, \cdots f_n\}$. Then $[\![P\ ;\ Q]\!] = [\![P]\!] \cup \bigcup_{i=1}^{n} [\![Q_i]\!]$ where Q_i are the copies of Q such that $f_i = i_Q$.

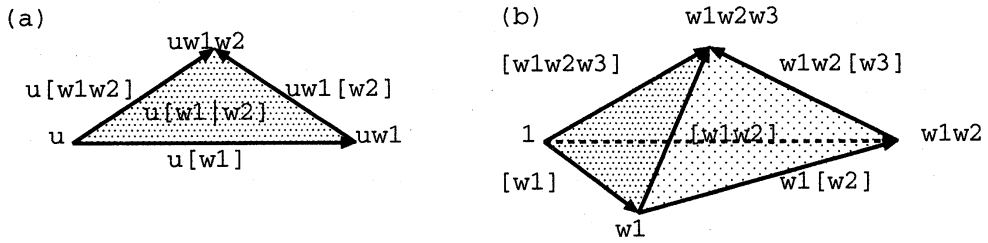
Restriction: $[\![P\backslash L]\!]$ is constructed as follows: First make $[\![P]\!]$, then remove all the cells representing the actions from L that do not communicate with their co-actions.

3 Bar Realization of the HDA Model

The HDA model can be formalized with tensor product notation as in [1]. However, we will use the bar expressions. There are two reasons for this. The first reason is that the computation paths can be directly represented in bar expression and this makes the analysis of the HDA model easier. The second reason is that we can use Bar construction in cohomology theory of groups. The parallelization algorithm and its justification rely on the properties of the universal covering space constructed by Bar construction. The bar expressions play the central role in the construction.

3.1 Bar expression

Let M be the free monoid over $\mathcal{A}$ with the unit word 1. An n-dimensional *bar expression* is the M-coefficient n-tuple $u[w_1|\cdots|w_n]$ $(u, w_i \in M)$ such that $u[w_1|\cdots|w_n] = 0$ if $w_i = 1$ for some i. This is in fact defined by the equation $u[w_1|\cdots|w_n] = (u, uw_1, uw_1w_2, \cdots, uw_1w_2\cdots w_n)$. This expression has a useful geometric interpretation. $u[w_1|\cdots|w_n]$ is the n-simplex with $n+1$ vertices $u, uw_1, uw_1w_2, \cdots, uw_1w_2\cdots w_n$. For any two vertices $v_1 = uw_1\cdots w_p$ and $v_2 = uw_1\cdots w_q$ $(p < q)$, $u[w_1|\cdots|w_n]$ has a unique directed edge from v_1 to v_2. The edge is labeled by $uw_1\cdots w_p[w_{p+1}w_{q+2}\cdots w_q]$. For any three vertices v_1, v_2 and $v_3 = uw_1\cdots w_r$ $(p < q < r)$, the n-simplex has a unique triangle face, i.e., 2-simplex, labeled by $uw_1\cdots w_p[w_{p+1}\cdots w_q|w_{q+1}\cdots w_r]$. The higher dimensional subsimplexes are constructed similarly. The following figure illustrates the examples of $n = 2$ and $n = 3$.



The boundary of the simplex is given by the boundary operator ∂_n (or simply ∂): $\partial_n = \partial_{0,1} + \partial_{1,n}$ where

$$\begin{aligned}\partial_{0,n}u[w_1|\cdots|w_n] &= \Sigma_{i=1}^{n-1}(-1)^i u[w_1|\cdots|w_{i-1}|w_iw_{i+1}|w_{i+2}|\cdots|w_n] + (-1)^n u[w_1|\cdots|w_{n-1}]\\ \partial_{1,n}u[w_1|\cdots|w_n] &= uw_1[w_2|\cdots|w_n]\end{aligned}$$

Notice that the boundary is represented with the formal sum of the edges with suitable sign $(+/-)$. For example, $\partial u[w_1|w_2] = uw_1[w_2] - u[w_1w_2] + u[w_1]$.

3.2 Cells in bar expression

A bar expression $u[w_1|\cdots|w_n]$ represents an n-simplex, but it is not an n-cell, which is really necessary in the HDA model. Hence we must construct an n-cell with n-simplexes by gluing them. First of all, we can define a *pseudo n-cell* $a_1 \otimes \cdots \otimes a_n$ as

$$a_1 \otimes \cdots \otimes a_n \stackrel{\text{def}}{=} \Sigma_{\sigma\in\mathcal{S}_n}\epsilon(\sigma)[a_{\sigma(1)}|\cdots|a_{\sigma(n)}]$$

where $\mathcal{S}_n$ is the n-dimensional symmetric group acting on the set of indices $1, 2, \cdots, n$. $\epsilon(\sigma)$ is the sign of the permutation $\sigma \in \mathcal{S}_n$.

As in [1], the boundary of a cell is divided into the *0-boundary*, which represents the initial part of the computation, and the *1-boundary*, which represents the final part of the computation. This division represents the direction of the computation. The 0/1-boundary operators on the pseudo n-cells are induced from the operators $\partial_{0,n}$ and $\partial_{1,n}$ for bar expressions:

$$\begin{aligned}\partial_{0,n}u(w_1\otimes\cdots\otimes w_n) &= u\Sigma_{1\le p<q\le n}(-1)^{q-1}(W^{(1)}_{p,q} - W^{(2)}_{p,q})\\ &\quad + u(\Sigma_{i=1}^{n}(-1)^i w_1\otimes\cdots\otimes w_{i-1}\otimes w_{i+1}\otimes\cdots\otimes w_n)\\ \partial_{1,n}u(w_1\otimes\cdots\otimes w_n) &= u(\Sigma_{i=1}^{n}(-1)^{i-1}w_i(w_1\otimes\cdots\otimes w_{i-1}\otimes w_{i+1}\otimes\cdots\otimes w_n))\end{aligned}$$

where $W^{(1)}_{p,q} = w_1\otimes\cdots w_{p-1}\otimes w_pw_q\otimes w_{p+1}\otimes\cdots\otimes w_{q-1}\otimes w_{q+1}\otimes\cdots\otimes w_n$ and $W^{(2)}_{p,q} = w_1\otimes\cdots w_{p-1}\otimes w_qw_p\otimes w_{p+1}\otimes\cdots\otimes w_{q-1}\otimes w_{q+1}\otimes\cdots\otimes w_n$. Notice that if the words $\{w_i\}_{1\le i\le n}$ satisfy the commutativity property with regard to word concatenation, i.e., $w_iw_j = w_jw_i$ for $1 \le i, j \le n$, then $u\Sigma_{1\le p<q\le n}(-1)^{q-1}(W^{(1)}_{p,q} - W^{(2)}_{p,q}) = 0$. This indicates that the pseudo n-cell

$u(w_1 \otimes \cdots \otimes w_n)$ represents the n-cell spanned by $w_1, \cdots, w_n$ if the commutativity property is satisfied. If we view the words $\{w_i\}$ as the names of the processes, the commutativity property means that any process in $\{w_i\}$ can run in any order so that the n-cell represents truly concurrent execution of these processes.

We now consider the example of the case of $n = 2$. If we assume the commutativity $ab = ba$, we obtain $\partial(a \otimes b) = \partial_0(a \otimes b) + \partial_1(a \otimes b) = ([a] - [b]) + (a[b] - b[a])$ and this is in fact the boundary of the 2-cell spanned by a and b. This situation is illustrated in Figure 2, which is in fact the formalized version of Figure 1(a). The 2-cell is obtained by gluing the bar expressions $[a|b]$ and $[b|a]$. Notice that the label of a vertex represents the path from the

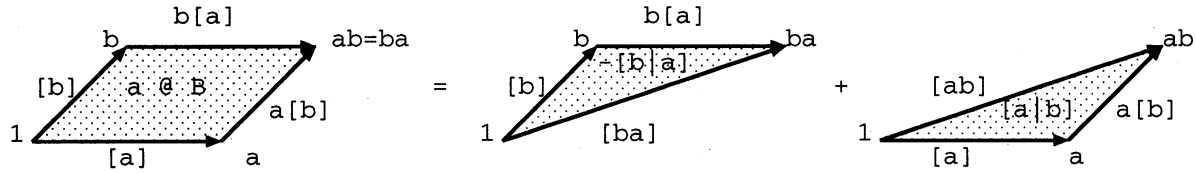


Figure 2: Construction of 2-cell with Bar expressions

base point 1 to the vertex. For example, if we can arrive at a vertex by passing the edges $[a_1], a_1[a_2], \cdots, a_1a_2 \cdots a_{n-1}[a_n]$ in this order, then the vertex is labeled by $a_1a_2 \cdots a_n$. The label of a vertex is unique in a given bar expression. However, the uniqueness is destroyed when bar expressions are glued. In the above example, one of the vertices in $a \otimes b$ has two labels ab and ba. This indicates that we can arrive at this vertex from the base point 1 in two ways: one is through the path $[a], a[b]$ and another is through the path $[b], b[a]$. These paths correspond to the computation paths (a, b) and (b, a) of $P = (a \mid b)$, and the vertex represents the final state of P, whose labels ab and ba indicate the paths.

We must also consider the treatment of the τ-action. Consider the process expression $P = (a \mid \bar{a})$. This may also be interpreted by the 2-cell $a \otimes \bar{a} = [a \mid \bar{a}] - [\bar{a} \mid a]$, but the synchronous communication $P \xrightarrow{\tau_a} (nil \mid nil)$ is not modeled explicitly. The 2-cell must contain the 1-cell that represents the τ_a-action. One solution of this problem is to introduce the new relation: $[ab] - [ba] = [\tau_a]$ if $b = \bar{a}$ and $[ab] - [ba] = 0$ otherwise. The idea behind this relation is as follows. The label $a\bar{a}$ represents a single state in which the actions a and $\bar{a}$ are finished. There are two ways to arrive at this state: a and $\bar{a}$ run in this order, or a and $\bar{a}$ run simultaneously carrying out the synchronous communication, namely the τ_a-action. The latter case is represented by the 1-cell $[a\bar{a}]$. On the other hand, by the similar discussion about the label $\bar{a}a$, we can regard that the 1-cell $[\bar{a}a]$ also represents the τ_a-action. Hence we can regard that the collection of the actions $[a\bar{a}]$ and $[\bar{a}a]$ represent a single τ_a-action. The equation $[a\bar{a}] - [\bar{a}a] = [\tau_a]$ represents this situation algebraically. The new relation allows the 2-cell $a \otimes \bar{a}$ to contain the 1-cell $[\tau_a]$, which occurs as the special part of the boundary of the cell. Namely, $\partial(a \otimes \bar{a}) = ([a] + a[\bar{a}] - \bar{a}[a] - [\bar{a}]) + [\bar{a}a] - [a\bar{a}] = ([a] + a[\bar{a}] - \bar{a}[a] - [\bar{a}]) - [\tau_a]$. We can understand that $[a] + a[\bar{a}] - \bar{a}[a] - [\bar{a}]$ part is the boundary in the ordinary sense, and that $[\tau_a]$ is the τ_a-action contained in the 2-cell $a \otimes \bar{a}$. Thus, the cell $a \otimes \bar{a}$ is illustrated in Figure 3, which is in fact the formalized version of Figure 1 (b).

Now we generalize this idea on the treatment of τ-actions. Let $E = u(w_1 \otimes \cdots \otimes w_n)$ be a pseudo n-cell. If we postulate (1) $W^{(1)}_{p,q} - W^{(2)}_{p,q} = w_1 \otimes \cdots \otimes w_{p-1} \otimes \tau_a \otimes w_{p+1} \otimes \cdots \otimes w_{q-1} \otimes w_{q+1} \otimes \cdots \otimes w_n$ if $w_p = \bar{w}_q = a$ or $\bar{w}_p = w_q = a$ for $1 \leq p < q \leq n$, and (2) the commutativity property $w_iw_j = w_jw_i$ for all i, j such that $w_i \neq \bar{w}_j$, then,

$$\begin{aligned} \partial(E) &= u(\Sigma_{i=1}^{n}(-1)^i w_1 \otimes \cdots w_{i-1} \otimes w_{i+1} \otimes \cdots \otimes w_n) \\ &\quad + u(\Sigma_{i=1}^{n}(-1)^{i-1} w_i(w_1 \otimes \cdots w_{i-1} \otimes w_{i+1} \otimes \cdots \otimes w_n)) + uR \end{aligned}$$

where $R = \Sigma(-1)^{q-1} w_1 \otimes \cdots \otimes w_{p-1} \otimes \tau_a \otimes w_{p+1} \otimes \cdots \otimes w_{q-1} \otimes w_{q+1} \otimes \cdots \otimes w_n$ and Σ ranges

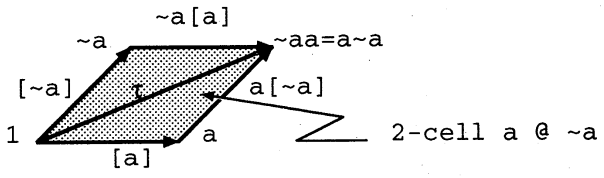


Figure 3: 2-cell containing τ-action

p and q such that $1 \leq p < q \leq n$ and $w_p = \bar{w}_q = a$ or $\bar{w}_p = w_q = a$ for some a.

3.3 Note on the equality between states

We postulated a kind of equality of words to make the simulated n-cells. The words to be equated are labels of the terminal states of the truly concurrent processes. In fact, this is not the algebraic equality. Hence we will call the equality the *merging relation* and denote by $\perp$ instead of $=$. For example, if $ab = ba$ holds, we wish to derive the new equality $wab = wba$ for arbitrary word w. Unfortunately, this is not accepted. For example, consider the process $P = (a \mid b) + c \; ; \; (a \; ; \; b + b \; ; \; a \; ; \; d)$. The actions a and b should be commutative in $(a \mid b)$, but the same actions should not be commutative in $c \; ; \; (a \; ; \; b + b \; ; \; a \; ; \; d)$: $cab \neq cba$. Hence, the right extension rule "If $U \perp V$, then $WU \perp WV$ (W is a word)" must be excluded. On the other hand, a simple observation will show that $\perp$ must satisfy symmetry, transitivity and the left extension rule "if $U \perp V$ then $UE \perp VE$" where U and V are words and E is a word or a bar expression. These rules are easily justified with the definition of bar expression.

3.4 HDA interpretation in the domain of bar expressions

Now we will reformulate the interpretation map $[\![\cdot]\!]$ given in section 2 in the domain of bar expressions. Because a unit square is not a primitive notion in our model, the interpretation of parallel composition is more complex than that given in [1]. We must introduce the $\star$-product for the construction of product spaces interpreting parallel composition. We will use the notations $ST = \{st \mid s \in S, t \in T\}$, $St = \{st \mid s \in S\}$, and $S \pm T = \{s \pm t \mid s \in S, t \in T\}$ for given sets S and T.

3.4.1 $\star$-product for parallel composition

$\star$-product is defined with the set of interleaving of elements in sequences. Let $x = a_1, a_2, \cdots, a_p$ and $y = a_{p+1}, a_{p+2}, \cdots, a_{p+q}$ be the sequences of length p and q. The set of all the patterns of interleave can be represented as $I(x,y) = \{a_{\sigma(1)}, a_{\sigma(2)}, \cdots, a_{\sigma(p+q)} \mid \sigma \in \mathcal{S}_{p+q}/\mathcal{S}_p\mathcal{S}_q\}$ where $\mathcal{S}_{p+q}/\mathcal{S}_p\mathcal{S}_q$ is the subset of $\mathcal{S}_{p+q}$ such that any element of $\mathcal{S}_{p+q}/\mathcal{S}_p\mathcal{S}_q$ does not change the order of $1, 2, \cdots, p$ and the order of $p+1, p+2, \cdots, p+q$. The set $I(x,y)$ will simply be called an interleave of two words x and y. For a set W of words, we define $I(x,W) \stackrel{\text{def}}{=} \bigcup_{w \in W} I(x,w)$. $I(W,x)$. $I(W_1, W_2)$ will be defined similarly.

Definition 1: ($\star$-product)
Let u, v, w and $\{w_i\}$ are words over $\mathcal{A}$. Then, the $\star$-product between bar expressions is a set of new bar expressions together with the set R of merging relations such that $w_1[u_1|\cdots|u_n] \star w_2[u_{n+1}|\cdots|u_{n+m}] = I(w_1,w_2)J([u_1|\cdots|u_n],[u_{n+1}|\cdots|u_{n+m}])$
where $J([u_1|\cdots|u_n],[u_{n+1}|\cdots|u_{n+m}]) = \Sigma_{\tau \in \mathcal{S}_{n+m}/\mathcal{S}_n\mathcal{S}_m} \epsilon(\tau)[u_{\tau(1)}|\cdots|u_{\tau(n+m)}]$, and the accompanied set of the merging relations are

$$R_{w_1[u_1\cdots],w_2[u_{n+1}\cdots]} = \begin{cases} \{w_1 \perp w_2 \mid w_1, w_2 \in I(w_1,w_2)\} & \text{if } n = m = 0 \\ \phi & \text{Otherwise.} \end{cases}$$

The following properties simplifies the calculation of product HDA space.

Proposition 1 *(Properties of $\star$-product)*
(1) $(E_1 \star E_2) \star E_3 = E_1 \star (E_2 \star E_3)$ *for bar expressions* E_i,
(2) $u([w_1] \star \cdots \star [w_n]) = \{u(w_1 \otimes \cdots \otimes w_n)\}$ *for words* u *and* w_i, *and*
(3) $w_1(u_1 \otimes \cdots \otimes u_n) \star w_2(v_1 \otimes \cdots \otimes v_m) = I(w_1, w_2)(u_1 \otimes \cdots \otimes u_n \otimes v_1 \otimes \cdots \otimes v_m)$
for words w_i, u_j, v_k.

3.4.2 Projection map φ_L for restriction

Let $L(\subset \mathcal{L})$ be a finite set. We will call the word in the form of $a\bar{a}$ or $\bar{a}a$ $(a \in L)$ the *communication of* a. In a bar expression, if the occurrence of an element from $\mathcal{A}$ is not contained in a communication of any symbol from L, it will be called the *isolated occurrence*. The projection map φ_L removes all the bar expressions that contains the isolated occurrences and replaces the communications such as $a\bar{a}$ and $\bar{a}a$ by the τ-action τ_a. For example, let H be the HDA space illustrated in Figure 3. Then, $\varphi_{\{a\}}(H)$ is the line segment $1 \xrightarrow{[\tau_a]} \tau_a$.

3.4.3 The interpretation map $[\![\cdot]\!]$

First of all, we define $com(S)$ procedure that generates the cells representing the synchronous communication.

Definition 2: ($com(S)$ procedure)
Let S be a set of bar expressions. Then the set $com(S)$ is constructed as follows: For arbitrary element $W \in S$ in the form of $W = u(a_1 \otimes \cdots \otimes a_m)$, if there exits $1 \leq p < q \leq n$ such that $\bar{a_p} = a_q$ and $\{a_p, a_q\} \cap \mathcal{L} = \{\alpha\}$, then construct a bar expression $E = u(a_1 \otimes \cdots \otimes a_{p-1} \otimes \tau_a \otimes a_{p+1} \otimes \cdots \otimes a_{q-1} \otimes a_{q+1} \otimes \cdots \otimes a_n)$ and make the new set $S \cup \{E\}$. $com(E)$ is the closure of this extension procedure.

Next we define the set of final states $F(P)$ for a process P. $F(P)$ is inductively defined as follows:

1. $F(nil) = \phi$, $F(a) = \{a\}$ and and $F(\tau_a) = \{a\bar{a}, \bar{a}a\}$ where $a \in \Sigma$;
2. If P and Q are process expressions, then $F((P \mid Q)) = F(P)F(Q) \cup F(Q)F(P)$, $F(P\ ;\ Q) = F(P)F(Q)$, $F(P + Q) = F(P) \cup F(Q)$ and $F(P\backslash L) = \varphi_L(F(P))$ for $L \subset \mathcal{L}$.

Then we can define the interpretation map $[\![\cdot]\!]$.

Definition 3: (HDA interpretation of process expressions)
Let P be a process expression. The *HDA interpretation* $[\![P]\!]$ of P together with the set $R_{[\![P]\!]}$ of the merging relations is defined inductively as follows:

1. $[\![nil]\!] = R_{[\![nil]\!]} = \phi$, $[\![a]\!] = \{1, a, [a]\}$ and $R_{[\![a]\!]} = \phi$, and $[\![\tau_a]\!] = \{1, a\bar{a}, \bar{a}a, [\tau_a]\}$ and $R_{[\![\tau_a]\!]} = \{a\bar{a} \perp \bar{a}a\}$ where $a \in \mathcal{A}$;
2. If P and Q are process expressions, then
   - $[\![(P \mid Q)]\!] = com\left(\bigcup_{f \in [\![P]\!], g \in [\![Q]\!]} f \star g\right)$ and $R_{[\![(P|Q)]\!]} = R_{[\![P]\!]} \cup R_{[\![Q]\!]} \cup \bigcup_{f \in [\![P]\!], g \in [\![Q]\!]} R_{f,g}$ where $R_{f,g}$ is the set of merging relations generated by the $\star$-product $f \star g$;
   - $[\![P\ ;\ Q]\!] = [\![P]\!] \cup F(P)[\![Q]\!]$ and $R_{[\![P\ ;\ Q]\!]} = R_{[\![P]\!]} \cup F(P)R_{[\![Q]\!]}$ where $F(P)R_{[\![Q]\!]} = \{fg \perp fh \mid f \in F(P), g \perp h \in R_{[\![Q]\!]}\}$;
   - $[\![P + Q]\!] = [\![P]\!] \cup [\![Q]\!]$ and $R_{[\![P+Q]\!]} = R_{[\![P]\!]} \cup R_{[\![Q]\!]}$
   - $[\![P\backslash L]\!] = \varphi_L([\![P]\!])$ and $R_{[\![P\backslash L]\!]} = \varphi_L(R_{[\![P]\!]})$

The following lemma claims that our HDA interpretation $[\![P]\!]$ is certainly a collection of the (simulated) n-cells.

Lemma 1 *For a process expression P, $[\![P]\!]$ has the decomposition into the disjoint union $[\![P]\!] = \bigcup_{i=0}^{n} [\![P]\!]_i$ for some n, where $[\![P]\!]_n \stackrel{\text{def}}{=} [\![P]\!] \cap \{u(w_1 \otimes \cdots \otimes w_n) \mid u, w_i \in M\}$. The number n will be called the dimension of $[\![P]\!]$ and denote by $n = dim[\![P]\!]$.*

Example 1: Consider the process expressions $P = (a \mid b \mid \bar{a})$ and $Q = P\backslash\{a\}$. Then,

$$\begin{aligned}[\![P]\!] = {} & \{1, a, \bar{a}, b, ab, ba, a\bar{a}, \bar{a}a, \bar{a}b, b\bar{a}, \bar{a}ab, a\bar{a}b, ab\bar{a}, \bar{a}ba, b\bar{a}a, ba\bar{a}\} \\ \cup {} & \{[a], [\bar{a}], [b], \bar{a}[a], b[a], a[\bar{a}], b[\bar{a}], a[b], \bar{a}[b], ab[\bar{a}], ba[\bar{a}], a\bar{a}[b], \bar{a}a[b], \bar{a}b[a], b\bar{a}[a], [\tau_a], b[\tau_a]\} \\ \cup {} & \{a \otimes b, \bar{a}(a \otimes b), a \otimes \bar{a}, b \otimes \bar{a}, a(b \otimes \bar{a}), b(a \otimes \bar{a}), \tau_a \otimes b\} \cup \{a \otimes b \otimes \bar{a}\}\end{aligned}$$

and $[\![Q]\!] = \varphi_{\{a\}}[\![P]\!] = \{1, b, \tau_a, \tau_a b, b\tau_a\} \cup \{[b], \tau_a[b], [\tau_a], b[\tau_a]\} \cup \{\tau_a \otimes b\}$.

3.4.4 Note on the occurrences of atomic actions

The same label may occur in a process expression more than once. For example, consider the processes $P = a + b \ ; (c \mid c)$ and $Q = a + b \ ; (c + ((a + c \ ; d) \mid e))$. In both cases, $[\![\cdot]\!]$ generates $b[c]$ twice according to the multiple occurrences of c. The naive set theoretic formulation of the interpretation cannot distinguish these $b[c]$s. We may avoid this problem by relabeling or attaching the occurrence numbers to the labels in the process expressions. However, for simplicity of the discussion, we will restrict the syntax so that this problem does not occur. We first define the *head action map zeta* inductively as follows: (1) $\zeta(\alpha) = \{\alpha\}$ where $\alpha \in \mathcal{A}$; (2) $\zeta(P + Q) = \zeta(P) \cup \zeta(Q)$; (3) $\zeta(P \ ; Q) = \zeta(P)$; (4) $\zeta((P \mid Q)) = \zeta(P) \cup \zeta(Q) \cup \Delta(P, Q)$ where $\Delta(P, Q) = \{\tau_a \mid l \in \zeta(P), \bar{l} \in \zeta(Q), l \in \mathcal{L} \cup \bar{\mathcal{L}}, l = a \text{ or } \bar{a}\}$; (5) $\zeta(P\backslash L) = \zeta(P) - L$. Then, we have the following.

Lemma 2 *Let P be a process expression. Then the HDA interpretation procedure does not generate duplicate bar expressions if and only if (1) for any subexpression $(Q \mid R)$, $\chi[Q] \cap \chi[R] = \phi$ and (2) for any subexpression $Q + R$, $\zeta(P) \cap \zeta(Q) = \phi$*

We will exclude the processes that does not satisfy the conditions in the above lemma. Because of the restriction of the syntax, the notion of strong bisimulation is drastically simple.

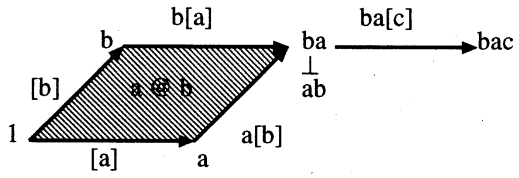
Lemma 3 *Two process expressions P and Q are bisimilar if and only if $[\![P]\!]_i = [\![Q]\!]_i$ for $i = 0$ and 1, where the merging relation between elements are not taken into account.*

4 Postulating Merging Relations

Consider, for example, process expressions $P = a \ ; b + b \ ; a$ and $Q = (a \mid b)$. $P \sim Q$ by the expansion law, so that the parallelization of P must be Q. Then, how should we find Q from P? The idea is that we regard the computation paths (a, b) and (b, a) in $cp(P)$ are homotopic, namely they can be continuously transformed to each other. For the two paths being homotopic, their end points must be equal and there must be the higher dimensional object in which the paths are continuously transformed. In our HDA model, the end points of the paths are ab and ba, so that we must first postulate the merging relation $ab \perp ba$. We will simply call this procedure *merging*. Then, we obtain a cycle that consists of the 1-cells $[a]$, $[b]$, $a[b]$ and $b[a]$. For the higher dimensional object that ensures homotopy relation, we fill the cycle with the 2-cell $a \otimes b$ to obtain a new HDA, which is in fact the HDA interpretation of Q.

In realizing this idea, we encounter the following problems: (1) How can we find the points to be merged? (2) How can we fill the cycles created by merging? (3) How can we obtain the parallelized process expressions from the new HDA spaces? This section is devoted to the first problem. The remainder will be considered in the sections 5 and 6.

Let P be a process. Then, the pairs of elements from $[\![P]\!]_0$ in the form of $< WAB, WBA >$ ($W \in M$, $A, B \in \chi[P]$) are called the *merging candidate pair* of P. We cannot merge all the merging candidate pairs. For example, consider a process expression $P = a\ ;\ b + b\ ;\ a\ ;\ c$. Then $[\![P]\!]_0 = \{1, a, b, ab, ba, bac\}$ and we obtain the candidate pair $< ab, ba >$. Assume that the candidate is merged and the cycle created with the merging is filled with the 2-cell $a \otimes b$, we obtain the HDA illustrated below. In fact, this is not the HDA interpretation of any process.



Hence, the candidate must be discarded.

Now we will see this example closely. The computation path (a, b, c) is not allowed in P, but it is allowed in the obtained HDA space. This means that this merging changes the observation behavior. Hence, we must check whether two computations beginning at a merging candidate pair are bisimilar. We introduce the *bisimilar condition* to check this. Based on Lemma 3, the condition is formally described as follows: Let P be a process expression. Then, we say that a merging candidate pair $< w_1, w_2 >$ satisfies the bisimilar condition if $[\![P(w_1)]\!]_i = [\![P(w_2)]\!]_i$ $(i = 0, 1)$ where $[\![P(w)]\!]_n = \{u[a_1|\cdots|a_n] \mid wu[a_1|\cdots|a_n] \in [\![P]\!]_n\}$.

5 Cell Construction in Bar Construction

In the parallelization procedure, we fill the cycles created by merging. The cycles are filled with the cells of various dimensions. We present here the systematic method of constructing the cells that fill the cycles. The cell construction is carried out in the framework of Bar construction.

5.1 Bar construction

Following [2], we will briefly explain (normalized) Bar construction for homology of monoid. Let P be a process expression and M be the free monoid over $\chi[P](\subset \mathcal{A})$. Let $\mathbf{Z}M$ be the monoid ring. Let B_n be the free $\mathbf{Z}M$-module over $M^n \stackrel{\text{def}}{=} \{[u_1|\cdots|u_n] \mid u_i \in M, u_i \neq 1\}$. Notice that B_n is also a $\mathbf{Z}$-module whose base is $\{u[w_1|\cdots|w_n] \mid u, w_i \in M, w_i \neq 1\}$. By identifying $u[]$ with u, we regard $B_0 = \mathbf{Z}M$. Then the sequence of modules

$$\cdots \xrightarrow{\partial_{n+1}} B_n \xrightarrow{\partial_n} \cdots \xrightarrow{\partial_2} B_1 \xrightarrow{\partial_1} \mathbf{Z}M \xrightarrow{\epsilon} \mathbf{Z} \longrightarrow 0$$

where $\epsilon(\Sigma_{n_i \in \mathbf{Z},\ m_i \in M} n_i m_i) = \Sigma n_i$, is called the Bar construction of M over $\mathbf{Z}$. In this sequence, a (n-)cycle is the element of $Ker\partial_n$. It is known that this sequence is exact. The geometric intuition of the Bar construction is as follows. We have the base $X_0 = \{w \mid w \in M\}$ of points labeled by M. Next, we construct the space $X_1 = \{u[w] \mid u, w \in M, w \neq 1\}$ of edges that link the points in X_0. Then we construct the space $X_2 = \{u[w_1|w_2] \mid u, w_i \in M, w_i \neq 1\}$ of 2-simplexes spanned by the edges from X_1 and so on. X_n interprets B_n and the Bar construction means the construction of the space $X = \bigcup_{n=0}^{\infty} X_n$. This space is in fact contractible, that is X has no cycles and is connected. Consider the n-skeleton $X^{(n)} = \bigcup_{i=0}^{n} X_i$. It is also known that $X^{(n)}$ does not have $(n-1)$-cycles. Because a HDA interpretation is represented in bar expression, the space can be embedded in X and we can find the cells that fill the cycles in X.

We do not need all part of the construction. Let Q be the parallelization of P. Consider the set of process expressions whose alphabet is $\chi[P]$. In this set, the process that has the largest degree of true concurrency is $(b_1 \mid \cdots \mid b_k)$ $(b_i \in \chi[P])$. Hence, the construction up to degree $\#\chi[P]$ is sufficient.

5.2 Cycles and constracting homptopy

We first define the constracting homotopy.

Definition 4: (Contracting homotopy)
Given a Bar resolution as in 5.1. Let $s_{-1} : \mathbf{Z} \longrightarrow B_0$ be the $\mathbf{Z}$-homomorphism such that $s_{-1}(1) = []$. Also for $n \geq 0$, let $s_n : B_n \longrightarrow B_{n+1}$ be the $\mathbf{Z}$-homomorphism such that $s_n(x[x_1 \mid \cdots \mid x_n]) = [x \mid x_1 \mid \cdots \mid x_n]$. $\{s_n\}$ is called the contracting homotopy of the resolution.

The contracting homotopy satisfies the equation $\partial_{n+1}s_n + s_{n-1}\partial_n = I_{B_n}$. If $C \in B_n$ is a cycle, $\partial_{n+1}s_n(C) + s_{n-1}\partial_n(C) = \partial_{n+1}s_n(C) = C$. This means that the map s_n makes the space $s_n(C)$ whose dimension is $dim\ C+1$ and whose boundary is C. In other words, s_n fills the cycle.

The cells constructed by the naive application of the contracting homotopy are not always preferable for our purpose. The following lemma indicates the way how we should use the map.

Lemma 4 *Let x be a cycle in B_n and let w be the largest common header in x, namely $x = wy$ and y has no common header. Then $ws_n(y) \approx s_n(x)$ (homotopic) in the universal covering space.*

Now we can give the cycle filling procedure. First we find cycles to be filled by calculating the homology group [1]. For a cycle $C = \Sigma_i E_i \in B_n$, let w be the largest common header of $\{E_i\}$: $C = w(\Sigma_i F_i)$ $(E_i = wF_i)$. Then $B = ws_n(\Sigma_i F_i)$ is the $(n+1)$-cell that fills the cycle C. Add all such cells to the given HDA space. New cycles can be generated in the new HDA. Hence, we calculate the homology group of the new HDA and repeat the application of the contracting homotopy. The application of the contracting homotopy is in fact the Bar construction procedure, and the loop of the cycle filling procedure terminates in finite steps as explained in the end of 5.1.

Example 2: Let $P = a\ ;\ (b \mid c) + b\ ;\ (a\ ;\ c + c\ ;\ a) + c\ ;\ (a \mid b)$. Then,

$$\begin{aligned}
[\![P]\!]_0 &= \{1, a, b, c, ab, ba, ac, ca, bc, cb, cab, cba, bac, bca, abc, acb\}, \\
[\![P]\!]_1 &= \{[a], [b], [c], b[a], c[a], a[b], c[b], a[c], b[c], bc[a], cb[a], ac[b], ca[b], ab[c], ba[c]\}, \\
[\![P]\!]_2 &= \{a(b \otimes c), c(a \otimes b)\}. \qquad R_{[\![P]\!]} = \{abc \perp acb, cab \perp cba\}
\end{aligned}$$

After merging, we obtain the new merging relations $R = \{ab \perp ba, ac \perp ca, bc \perp cb, bac \perp bca\}$. Now we have four 1-cycles $C_1 = [a] + a[b] - b[a] - [b]$, $C_2 = [a] + a[c] - c[a] - [c]$, $C_3 = [b] + b[c] - c[b] - [c]$ and $C_4 = b[a] + ba[c] - b[c] - bc[a] = b([a] + a[c] - [c] - c[a])$. The 2-cells that fill these cycles are constructed with the contraction homotopy $s_1 : B_1 \rightarrow B_2$: $s_1C_1 = [a \mid b] - [b \mid a] = a \otimes b$, $s_1C_2 = [a \mid c] - [c \mid a] = a \otimes c$, $s_1C_3 = [b \mid c] - [c \mid b] = b \otimes c$, and, for C_4, $bs_1([a] + a[c] - [c] - c[a]) = b([a \mid c] - [c \mid a]) = b(a \otimes c)$ by Lemma 4. Then we extend the HDA by adding these 2-cells to obtain the new HDA, $HDA_1 = [\![s]\!] \cup \{a \otimes b, a \otimes c, b \otimes c, b(a \otimes c)\}$. Now HDA_1 has a 2-cycle $C_5 = -a \otimes b + a \otimes c - b \otimes c + a(b \otimes c) - b(a \otimes c) + c(a \otimes b)$ and the 3-cell that fills this cycle is $s_2(C_5) = [a \mid b \mid c] - [a \mid c \mid b] - [b \mid a \mid c] + [b \mid c \mid a] + [c \mid a \mid b] - [c \mid b \mid a] = a \otimes b \otimes c$. Then we extend HDA_1 by adding this 3-cell to obtain the new HDA,

$$\begin{aligned}
HDA_2 &= HDA_1 \cup \{a \otimes b \otimes c\} = \\
&\{1, a, b, c, ab, ba, ac, ca, bc, cb, cab, cba, bac, bca, abc, acb\},
\end{aligned}$$

$\cup$ $\{[a],[b],[c],b[a],c[a],a[b],c[b],a[c],b[c],bc[a],cb[a],ac[b],ca[b],ab[c],ba[c]\}$,
$\cup$ $\{a\otimes b, a\otimes c, b\otimes c, b(a\otimes c), a(b\otimes c), c(a\otimes b)\}\cup\{a\otimes b\otimes c\}$

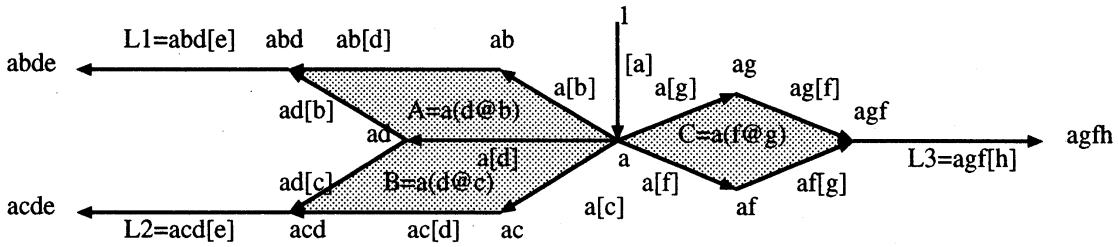
This is in fact acyclic and the cell construction procedure finishes. Notice that HDA_2 is certainly $[\![(a \mid b \mid c)]\!]$.

6 Reverse Interpretation

In the parallelization algorithm, we construct the new HDA spaces from the HDA interpretations. The final stage of the algorithm is to extract the parallelized processes from the new HDA spaces. We will call this step the *reverse interpretation*. Notice that any HDA space, even the HDA spaces obtained by the cycle filling, is not always the HDA interpretation of a process expression. For example, consider a process $P = a\ ;\ b + b\ ;\ (a + c)$. We obtain a HDA space $H_{wrong} = \{1, a, b, ab, ba, bc\} \cup \{[a],[b],a[b],b[a],b[c]\} \cup \{a \otimes b\}$ with the new merging relation $ab \perp ba$, and H_{wrong} does not interpret any process expression. The basic idea of the reverse interpretation is that the HDA interpretation of a process expression consists of the higher dimensional computation paths (HDCP) and the HDCPs may branch and intersect according to the nature of the given process.

6.1 HDCP searching

We will first define the notion of higher dimensional computation paths (HDCP). In a HDA space we consider the maximal cells according to the partial ordering defined by set inclusion. A *higher dimensional computation path* (HDCP) is a sequence of the maximal cells. For example, consider the HDA space given in the following figure which consists of four maximal 1-cells $[a]$, L_1, L_2 and L_3, and three maximal 2-cells A, B and C.



The HDCPs are $s_1 = ([a], A, L_1)$, $s_2 = ([a], B, L_2)$ and $s_3 = ([a], C, L_3)$. We can interpret a HDCP as a fragment of concurrent computation. For example, s_1 means that first a runs, after that b and d run truly concurrently and finally e runs.

Now we will give the rule by which the maximal cells are arranged in a HDCP. Consider the HDCP s_1 in the above example. The 1-boundary of $[a]$ is a and it is at the same time a part of the 0-boundary of A. In fact, $\partial_0(A) = a[b] - a[d]$ and $\partial_0(a[b]) = \partial_0(a[d]) = a$. Hence we can regard that $a \in \partial_0(A)$. On the other hand, the 0-boundary of L_1 is $abd(= \partial_0(L_1))$ and this is at the same time a part of the 1-boundary of A. Recall that the 0-boundary represents the initial part of the subprocess and the 1-boundary represents the final part of the subprocess. Then, the arrangement of the cells in a HDCP gives the direction of the computation. Assume the maximal cells C_1 and C_2. If $C_1 \cap C_2$ is the cell which is a part of the 1-boundary of C_1 and a part of the 0-boundary of C_2, we will denote it by $C_1 \Rightarrow C_2$. We will call '$\Rightarrow$' the *next-to relation* and a HDCP is a maximal sequence, $C_1 \Rightarrow \cdots \Rightarrow C_n$, of maximal cells ordered by the next-to relation such that the start point of C_1 is 1.

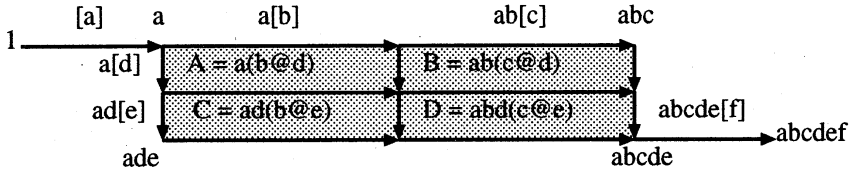
Now the HDCP searching procedure is abstractly described as follows: **[Step 1]** Construct the set SE of maximal sequences $C_1 \Rightarrow \cdots \Rightarrow C_n$ in a given HDA space, **[Step 2]** If there is

a sequence $C_1 \Rightarrow \cdots \Rightarrow C_n \in SE$ such that the start state of C_1 is not 1, then HDCP search fails. Otherwise, output SE. Notice that HDCP search fails in Step 2 for the HDA space H_{wrong}.

Proposition 2 *For arbitrary process expression P, (1) the HDCP search procedure succeeds for $[\![P]\!]$ and (2) the set of HDCPs covers $[\![P]\!]$, namely every cell in $[\![P]\!]$ is contained in a maximal cell in one of the HDCPs.*

6.2 Interleaving search

A set of HDCPs may represents an interleaving subprocess. For example, consider a HDA interpretation $H = [\![a\ ;\ ((b\ ;\ c)|(d\ ;\ e))\ ;\ f]\!]$ in the following figure. We find three HDCPs s_i



($i = 1 \sim 3$) in H: $s_1 = [a] \Rightarrow A \Rightarrow B \Rightarrow D \Rightarrow abcde[f]$, $s_2 = [a] \Rightarrow A \Rightarrow C \Rightarrow D \Rightarrow abcde[f]$, $s_3 = [a] \Rightarrow A \Rightarrow D \Rightarrow abcde[f]$. The cluster of the cells A, B, C and D forms the interpretation of the subprocess $((b\ ;\ c)|(d\ ;\ e))$. We can also observe that s_1 and s_2 suffice to find the cells in the cluster and these HDCPs are different only at the third element, B and C. We first generalize this nature of s_1 and s_2 to obtain the notion of pre-interleaving model.

Definition 5: (Homotopy relation of HDCP)
Let $s = C_1 \Rightarrow \cdots \Rightarrow C_m$ and $t = D_1 \Rightarrow \cdots \Rightarrow D_n$ be HDCPs. Then, s and t are called *adjacent* iff $m = n$, and there exists at most one index i_0 such that $1 < i_0 < m$ and $C_i = D_i$ for arbitrary i ($\neq i_0$). The *homotopy relation*, $\approx$, between HDCPs is the reflexive, symmetric and transitive closure of the adjacent relation.

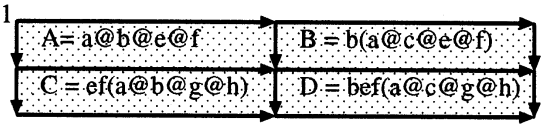
Definition 6: (Pre-interleaving model)
Let $S = \{s_i\}_{1 \leq i \leq m}$, where $s_i = C_{i,1} \Rightarrow C_{i,2} \Rightarrow \cdots \Rightarrow C_{i,n(i)}$, be a subset of HDCPs in a given HDA. Then, S is called a *pre-interleaving model* iff (1) $s_i \approx s_j$ for arbitrary $1 \leq i, j \leq m$, hence $n(i)$ is a constant L, (2) there exist p and q ($1 \leq p < q \leq L$) such that for arbitrary $1 \leq k \leq p$ and $q \leq l \leq L$, $C_{i,k}$ and $C_{i,l}$ are constants for arbitrary i, and (3) for p and q in (2), $dim(C_{i,p-1} \cap C_{i,p}) = dim(C_{i,l} \cap C_{i,l+1}) = 0$ for all i.

A pre-interleaving model may not contain sufficient number of cells to intepret an interleaving process. In the above example, $\{s_2\} (\subset \{s_1, s_2\})$ is a pre-interpretation model, but we cannot find sufficient number of cells in the model. On the other hand, the pre-interleaving model $\{s_1, s_2\}$ covers sufficient number of maximal cells. Thus, we introduce the notion of interleaving model, which has sufficient cells in the cluster. Before that we define the notion of sequential component system (SCS) of a pre-interleaving model.

For simplicity, we focus our attention to the cell clusters and we assume p and q in Def. 6 equal to 1 and L. Let $S = \{s_i\}_{1 \leq i \leq m} = \{C_{i,1} \Rightarrow \cdots \Rightarrow C_{i,L} \mid C_{i,j} = w_{i,j}(x_{i,j}^1 \otimes \cdots \otimes x_{i,j}^{n(i,j)})\}_{1 \leq i \leq m}$ be a pre-interleaving model. Assume that there exists a natural number $N \geq 2$, such that, for all i and j, $\overline{x}_{i,j} = (x_{i,j}^1, \cdots, x_{i,j}^{n(i,j)})$ is divided to nonempty subsequences $\overline{x}_{i,j} = (\overline{\alpha}_{i,j}^1, \cdots, \overline{\alpha}_{i,j}^N)$ carrying out permutation of elements if necessary. We also assume that (1) for arbitrary i there exists only one index k_i such that for arbitrary j, $\overline{\alpha}_{i,j}^k = \overline{\alpha}_{i,j+1}^k$ if $k \neq k_i$ and (2) for arbitrary i and j if $w_{i,j} = a_1 \cdots a_p$ and $w_{i,j+1} = b_1 \cdots b_q$ then there exists k such that $\overline{\alpha}_{i,j}^k = (c_1, \cdots, c_r)$

where $\{c_1, \cdots, c_r\} = \{b_1, \cdots, b_q\} - \{a_1, \cdots, a_p\} (\subset \{x^1_{i,j}, \cdots, x^{n(i,j)}_{i,j}\})$. In this case, we will simply say that the *partition* of the pre-interleaving model is given. Also we will call the number N the *partition number*.

Example 3: Consider the cell cluster given in the following figure. We have a pre-interleaving



model $\{s_1\} = \{A \Rightarrow C \Rightarrow D\}$, and and $\overline{x}_{1,1} = (a, b, e, f)$, $w_{1,1} = 1$, $\overline{x}_{1,2} = (a, b, g, h)$, $w_{1,2} = ef$, $\overline{x}_{1,3} = (a, c, g, h)$ and $w_{1,3} = bef$. Then we have two candidates of partitions of the cell cluster.

$\bar{x}_{1,1} = (\bar{\alpha}^1_{1,1}, \bar{\alpha}^2_{1,1}) = ((a, b), (e, f))$, $\bar{x}_{1,1} = (\bar{\alpha}^1_{1,1}, \bar{\alpha}^2_{1,1}, \bar{\alpha}^3_{1,1}) = ((a), (b), (e, f))$
$\bar{x}_{1,2} = (\bar{\alpha}^1_{1,2}, \bar{\alpha}^2_{1,2}) = ((a, b), (g, h))$, and $\bar{x}_{1,2} = (\bar{\alpha}^1_{1,2}, \bar{\alpha}^2_{1,2}, \bar{\alpha}^3_{1,2}) = ((a), (b), (g, h))$
$\bar{x}_{1,3} = (\bar{\alpha}^1_{1,3}, \bar{\alpha}^2_{1,3}) = ((a, c), (g, h))$, $\bar{x}_{1,3} = (\bar{\alpha}^1_{1,3}, \bar{\alpha}^2_{1,3}, \bar{\alpha}^3_{1,3}) = ((a), (c), (g, h))$

The first partition is the case of $N = 2$ and the second is the case of $N = 3$. Both cases satisfy the condition (1) above. Next consider the condition (2). The partition of the sequence (a, b, e, f) must contain the subsequence (e, f). Also, the partition of (a, b, g, h) must contain the subsequence (b), which is obtained by $\{b, e, f\} - \{e, f\}$. Hence, we can accept only the case of $N = 3$.

A partition of a pre-interleaving model induces the partition of a HDCP in the model. The partition of a HDCP $s_i = C_{i,1} \Rightarrow \cdots \Rightarrow C_{i,L}$ can be conveniently represented by the $(L \times N)$-matrix PM_i which will be called the *partition matrix*. PM_i is defined as $PM_i = [E_{k,l}]_{k,l}$ where $E_{k,l} = \bar{\alpha}^l_{i,k}$ and $\bar{x}_{i,j} = (\bar{\alpha}^1_{i,j}, \bar{\alpha}^2_{i,j}, \cdots, \bar{\alpha}^N_{i,j})$. Then, by removing duplicated elements in each column and with suitable renaming of indices, we obtain the vectors: $A^i_1 = {}^t[\bar{\alpha}^1_{i,1}, \bar{\alpha}^1_{i,2}, \cdots, \bar{\alpha}^1_{i,\varphi(1)}]$, $A^i_2 = {}^t[\bar{\alpha}^2_{i,1}, \bar{\alpha}^2_{i,2}, \cdots, \bar{\alpha}^2_{2,\varphi(2)}]$, $\cdots$, $A^i_n = {}^t[\bar{\alpha}^n_{i,1}, \bar{\alpha}^n_{i,2}, \cdots, \bar{\alpha}^n_{n,\varphi(n)}]$, where we used the transposition ${}^t(\cdot)$ simply for the typographical reason. We will omit the transposition in the following. Then, we will call $\{A^i_k\}_{1 \le k \le n}$ the *sequential component system* (SCS) with regard to s_i. We will interpret A^i_j as the sequential composition $seq_j = P_1 \ ; \ \cdots \ ; \ P_{\varphi(j)}$ where $P_p = (a^p_1 \mid \cdots \mid a^p_k)$ and $\bar{\alpha}^j_{i,p} = (a^p_1, \cdots, a^p_k)$. Then the process expression $(seq_1 \mid \cdots \mid seq_n)$ will be called the process interpreting the partition of the HDCP.

Example 4: Consider the cell cluster explained in the previous example. We consider the HDCP s_1 and s_2. We obtain the following partition matrices.

$$PM_1 = \begin{bmatrix} \bar{\alpha}^1_{1,1} & \bar{\alpha}^2_{1,1} & \bar{\alpha}^3_{1,1} \\ \bar{\alpha}^1_{1,2} & \bar{\alpha}^2_{1,2} & \bar{\alpha}^3_{1,2} \\ \bar{\alpha}^1_{1,3} & \bar{\alpha}^2_{1,3} & \bar{\alpha}^3_{1,3} \end{bmatrix} = \begin{bmatrix} (a) & (b) & (e, f) \\ (a) & (b) & (g, h) \\ (a) & (c) & (g, h) \end{bmatrix} \qquad PM_2 = \begin{bmatrix} (a) & (b) & (e, f) \\ (a) & (c) & (e, f) \\ (a) & (c) & (g, h) \end{bmatrix}$$

For s_1, the SCS is $A^1_1 = [(a)]$, $A^1_2 = [(b), (c)]$ and $A^1_3 = [(e, f), (g, h)]$. Then the process interpreting the partition is $(seq_1 \mid seq_2 \mid seq_3) = (a \mid (b \ ; \ c) \mid (e \mid f) \ ; \ (g \mid h))$. For s_2, we obtain the same SCS and the process expression.

The SCS is determined uniquely when the partition is given. On the other hand, given a HDCP the partition is not always determined uniquely. For the uniqueness of the process interpreting the partition, we have the following result.

Lemma 5 *Given a HDCP in a pre-interleaving model, the obtained process expression is uniquely determined regardless of the partition of the model.*

Now we introduce the notion of interleaving models. A pre-interleaving model may contain sufficient number of cells to form a cell cluster that interprets an interleaving process. The number of cells needed can be given by a simple combinatorial reasoning. On the other hand, the process expression interpreting a partition is in fact the candidate for the interleaving process. Hence, if all the process interpreting the partition, which are determined for each HDCP in a model, are equal, then the model may be the interpretation of the process. An interleaving model is the pre-interleaving model that contains sufficient number of cells and the common process expression interpreting the partitions.

Definition 7: (Restricted version of Interleaving model)
Let $S = \{s_i\}_{1 \leq i \leq m}$ be a pre-interleaving model and P be a process expression. Then, S is called the *interleaving model* of P iff (1)P is the common process interpreting the partitions of all s_i, (2) $L = \Sigma_{i=1}^{n} \varphi(i) - n + 1$, and (3) $m = (\Sigma_{i=1}^{n}(\varphi(i) - 1))! / \Pi_{i=1}^{n}(\varphi(i) - 1)!$, where L, n and $\varphi(i)$ are the common length of s_i, the partition number and the length of the i-th element of the SCS.

Now we return to the general setting of p and q in Def. 6. Let $\{s_i\}_{1 \leq i \leq m}$, where $s_i = C_{i,1} \Rightarrow \cdots \Rightarrow C_{i,L}$, be a pre-interleaving model with the index p and q as in Def. 6. If $S' = \{C_{i,p} \Rightarrow \cdots \Rightarrow C_{i,q}\}_{1 \leq i \leq m}$ satisfies the condition given in Def. 7, we will call $\{s_i\}_{1 \leq i \leq m}$ the *interleaving model* with the common process $P = (seq_1 \mid \cdots \mid seq_n)$.

Then, the interleaving search procedure goes as follows. Given an interleaving model as above. We will make the new HDCP $C_{1,1} \Rightarrow \cdots C_{1,p-1} \Rightarrow (seq_1 \mid \cdots \mid seq_n) \Rightarrow C_{1,q+1} \Rightarrow \cdots \Rightarrow C_{1,L}$. After that we remove all the HDCPs passing through the cell cluster represented by S'. For example, the HDCPs s_i $(i = 1, 2, 3)$ given in the beginning of this subsection pass through the cell cluster, and we obtain the new HDCP $[a] \Rightarrow ((b \ ; \ c) \mid (d \ ; \ e)) \Rightarrow [f]$ in this case.

If there exists a pre-interleaving model that is not contained in any interleaving model, the interleaving search will fail. For example, if the maximal cell B lacks in the cell cluster given in the beginning of this subsection, the HDA is not the interpretation of any process expression.

6.3 Reverse interpretation algorithm

We will give the algorithm *revInt* of reverse interpretation. This algorithm may fail. If it fails, the HDA space does not interpret any process expressions and this means that the given process expression cannot be parallelized. The basic idea of *revInt* is that the HDCPs after the interleaving search represent nondeterministic branches so that $\Rightarrow$ can be replaced by the sequential composition, each cell $C = W(E_1 \otimes \cdots \otimes E_n)$ can be replaced by the parallel composition $W(E_1 \mid \cdots \mid E_n)$, and the HDCPs can be combined with the nondeterministic choice. However, the situation is a little more complex because the HDCPs may intersect at the cells other than 1. For example, consider the HDA spaces $H_1 = [\![a \ ; \ ((b \mid c) + d)]\!]$ and $H_2 = [\![((a + b) \mid c)]\!]$. We can find the set of HDCPs $s_1 = a \otimes d$ and $s_2 = a \otimes b \otimes d$ for H_1 and $t_1 = a \otimes c$ and $t_2 = b \otimes c$ for H_2. Notice that $s_1 \cap s_2 = \{[a]\}$ and $t_1 \cap t_2 = \{[c]\}$. Then we cannot make the process $(a \mid d) + (a \mid b \mid d)$ from s_i nor the process $(a \mid c) + (b \mid c)$ from t_i. Hence, we must introduce a set of rewrite rules to handle this problem.

Algorithm: *revInt*

1. Let SE be the set of HDCPs obtained by the HDCP search procedure followed by the interleaving search procedure. If the interleaving search fails, the hole procedure fails.
2. For each HDCP $s = C_1 \Rightarrow \cdots \Rightarrow C_n$ in SE, replace '$\Rightarrow$' by ' ; ' and replace '$\otimes$' (in C_i) by '|'. Let PP be the set of process expressions obtained by this translation.
3. Make the expression $Exp \stackrel{\text{def}}{=} \Sigma_{P \in PP} P$.
4. Translate Exp with the rewrite rules given below.

Rule 1: $(P\ ;\ (X \mid Y_1)\ ;\ \cdots\ ;\ (X \mid Y_k)\ ;\ Q) + (P\ ;\ (X \mid Z_1)\ ;\ \cdots\ ;\ (X \mid Z_l)\ ;\ R)$
$\longrightarrow P\ ;\ (X \mid ((Y_1\ ;\ \cdots\ ;\ Y_k) + (Z_1\ ;\ \cdots\ ;\ Z_l)))\ ;\ Q$
Rule 2: $(P\ ;\ Q) + (P\ ;\ R) \longrightarrow P\ ;\ (Q + R)$
Rule 3: $(P\ ;\ S) + (Q\ ;\ S) \longrightarrow (P + Q)\ ;\ S$
Rule 4: $\tau_a \longrightarrow (a \mid \bar{a})\backslash\{a\}$

If a redex of Exp matches the left hand side of Rule 1, $revInt$ fails unless $Q = R$. If a redex of Exp matches the left hand side of Rule 2 and $\zeta(Q) \cap \zeta(R) \neq \phi$, $revInt$ fails unless Rule 1 can be applied to $Q + R$ part.

Example 5: Consider again the HDA space in the beginning of 6.1.

$$\begin{aligned} Exp &= s_1 + s_2 + s_3 \\ &= (a\ ;\ (d \mid b)\ ;\ e) + (a\ ;\ (d \mid c)\ ;\ e) + (a\ ;\ (f \mid g)\ ;\ h) \\ &= (a\ ;\ (d \mid (b + c))\ ;\ e) + (a\ ;\ (f \mid g)\ ;\ h) \qquad \text{by Rule 1} \\ &= a\ ;\ ((d \mid (b + c))\ ;\ e + (f \mid g)\ ;\ h) \qquad \text{by Rule 2} \end{aligned}$$

Then we know that the HDA is $[\![a\ ;\ ((d \mid (b + c))\ ;\ e + (f \mid g)\ ;\ h)]\!]$.

The following theorem claims the fundamental property of the parallelization procedure.

Theorem 1 *Let P be a process expression. Let $genHDA$ be the procedure that generates new HDA spaces by merging and the cell construction, and let $par(P) \stackrel{\text{def}}{=} revInt(genHDA(P))$. Then (1) If $par(P)$ has a value, $P \sim par(P)$ and $par(par(P)) = par(P)$, (2) The parallelization is compatible with the expansion law: $(\bar{P}_1 \mid \cdots \mid \bar{P}_n)\backslash L = revInt(genHDA(\Sigma\{\alpha\ ;\ (P_1 \mid \cdots \mid Q_i \mid \cdots \mid P_n)\backslash L\} + \Sigma\{\tau_a\ ;\ (P_1 \mid \cdots \mid Q_i \mid \cdots \mid Q_j \mid \cdots \mid P_n)\backslash L\}))$ where $\bar{P}_i$ is the parallelized version of P_i, (3) $revInt([\![P]\!]) = P$.*

7 Conclusion

The HDA model of true concurrency gives a geometrically simple idea for parallelization algorithm: Find the states to be merged and fill the cycles to obtain the HDA interpretation of the parallelized process. We showed that the algorithm has close relation with Bar construction and combinatorial geometry. Unfortunately, we could not present the proofs of propositions or the extensive discussion on subtle issues because of the limitation of the space. See [5] for the detail

References

[1] E. Goubault and T. P. Jensen. Homology of Higher Dimensional Automata. In *3rd International Conference on Concurrency Theory, LNCS 630.* Springer–Verlag, 1992.

[2] S. S. Mac Lane. *Homology.* Springer–Verlag, 1963.

[3] V. Pratt. Modeling Concurrency with Geometry. In *18th Annual ACM Symposium on Principles of Programming Languages.* ACM, 1991.

[4] E. H. Spanier. *Algebraic Topology.* McGraw-Hill, 1966.

[5] Y. Takayama. Bar Construction as Parallelization in Process Algebra, 1995. preprint.